https://doi.org/10.2352/ISSN.2169-4451.2017.32.221
©2016; Society for Imaging Science and Technology

# Development of a Supervision System: Towards Closing the Control Loop in 3D Printing Systems

*Alvaro J. Rojas Arciniegas[1,3*], Juan C. Amaya Hurtado[2,3]; [1]Department of Automatics and Electronics, [2]Mechatronics Engineering, [3]Research Group in Technologies for Manufacture (GITEM), College of Engineering, Universidad Autonoma de Occidente, Cali, Colombia. [*]Corresponding author: ajrojas@uao.edu.co*

## Abstract

*During the 3D printing process errors can occur and go unnoticed until it reaches the user due to the lack of closed loop control. The low-end FFF printers usually suffer more of these failures caused by absence of material or movement of the filament interrupted by different reasons: The filament spool gets tangled causing that the filament cannot reach the extrusion nozzle, the motor that feeds the extruder stop working, or an external object is blocking the movement of the filament. To address these, the work reported takes the first steps towards closing the loop: supervising the extrusion nozzle, detecting material flowing out, and correlating it to the printing process to determine if it is occurring when it should. The supervision is performed with a camera pointing at the nozzle and an interface has been developed to select one of 10 common filament colors that can be monitored. The system then process the image captured and identifies only the filament as it's coming out of the nozzle, enabling to detect discontinuities on the material flow. Although the system is being developed for FFF printers, the need for feedback and closing the loop on the printing process would benefit all additive manufacture processes; therefore, the work proposed is seen as a test case that can later be scaled to other printing technologies.*

## Introduction

3D printing consists in the successive addition of material by layers, allowing creating three-dimensional objects with complex geometries from a digital file. There are many types of 3D printers relying on different technologies, different methods to print, and use different materials, but they all share the same principle. Some of the common 3D printing technologies are:

- Stereolithography (SLA): uses liquid UV curable photopolymer and a UV laser to build each layer [1].
- Binder Jetting: Uses two materials, a powder material and a liquid binder that is printed to bond the powder particles, forming each layer of the desired geometry.
- Material Jetting: it uses a support material and a build material; both materials are applied in droplets by an inkjet printhead. After the material is printed, a UV light cures the layers and hardens it.
- Powder Bed Fusion: This method fuses small particles of material (Metal, plastic, ceramic or glass powders) using a power laser to create the desired final object [2].
- Directed Energy Deposition: Consists in the deposition of metal powder on a surface where a laser or electron beam fuses the material, creating the layers of the final object.
- Fused Filament Fabrication (FFF): This process uses filament of thermoplastics materials (such as ABS and PLA) that passes through a heated extrusion nozzle; the material melts and is deposited on the platform surface to recreate each cross section of the 3D model [3].

A relative movement between the extrusion nozzle / print head / fusing laser and the building platform is created through a numerically controlled mechanism. Local controllers intend to guarantee that the desired relative movements are achieved to produce the appropriate layer and ultimately the 3D structure; however, the vast majority of 3D printers work in open loop, meaning there is no feedback that ensures that the printing process is occurring as intended. In fact, controls for additive manufacture has been identified as one of the key areas required for further development of additive manufacturing technologies [4].

## Printing Challenges in FFF

3D printing systems are far from being perfect and the vast majority do not work with closed loop control. The low-end printers are usually FFF systems and they could suffer from multiple failures in the fabricating process (generally more than high-end systems) that usually go unnoticed until it reaches the user with the defective printed object, making them good test subjects for this study. These systems rely on subsystem-level closed loop controls of temperature for the nozzle and platform, and the accurate positioning of the motors, to reproduce the intended geometry. Some of the challenges in FFF 3D printing systems are:

- The filament spool gets tangled causing that the filament does not reach the head extruder.
- The filament slips within the extruder during the printing process causing discontinuities and undesired gaps on the printed part.
- The extrusion nozzle gets clogged.
- The printer runs out of filament or material and it goes undetected for the printer.
- The material keeps flowing out of the nozzle in undesired locations.
- Miss-registration between layers that causes imperfections on the reproduced geometry of the object, particularly in terms of surface finish.
- The printing material sticks to the outside of the nozzle, disturbing previously printed layers.
- The printed part does not adhere well to the building platform.
- The filament does not attach to the previous layer.

These failure modes can be summarized in three basic problems: 1. Material flows out of the nozzle when it should not or it does not occur when it should; 2. The relative movement of the nozzle and building platform is not being performed accurately; and 3. There are issues with the partially printed structure on the building platform. Particularly for the first type of failures, the filament is usually not being pushed by the driving gear into the extruder causing this material to exit the nozzle in a discontinuous/uncontrolled manner (see **Figure 1**).

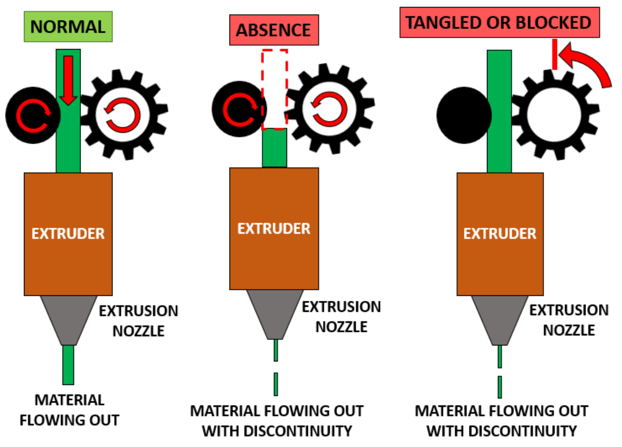


***Figure 1.*** *Left: Normal FFF printing process; Middle: printing error due to absence of filament; and Right: printing error due to tangled or blocked filament.*

To address these, some strategies have been envisioned: 1. Supervise the material output of the nozzle to provide feedback and prevent the fabrication process to continue if failures have occurred; 2. Sense the relative position of the nozzle and platform and close the loop on the positioning sub-system to account for losing steps (most of the motors used are stepper motors) or errors in the transmissions used (belts, pulleys, screws); and 3. Supervise the printed part as it is being printed to automatically detect abnormalities on the building platform [5].

The work reported address the first strategy towards closing the loop: supervising the extrusion nozzle, detecting material flowing out, and correlating it to the printing process to determine if it is occurring when it should. The supervision is performed with a camera pointing at the nozzle and an interface has been developed to select one of 10 common filament colors that can be monitored. The system then process the image captured, and identifies only the filament as it's coming out of the nozzle, enabling to detect discontinuities on the material flow. Although the system is being developed for FFF printers, the need for feedback and closing the loop on the printing process would benefit all additive manufacture processes [4, 6, 7]; therefore, the work proposed is seen as a test case that can later be scaled to other printing technologies.

## Development of the supervision system

The development of the supervision system was made by dividing the problem in several tasks: capturing images, defining the color to monitor out of the ones that are normally used on 3D printing filaments (Red, Green, Yellow, Orange, Black, White, Blue, Cyan, Violet and Magenta), monitor the material flow out of the nozzle which implies detecting only the material flow in the image, and then detecting discontinuities to correlate with the G-code to determine whether it is intended or not. For the development of the application it was decided that python with OpenCV would be used in order to transfer it to embedded platforms such as Raspberry Pi. The details of each phase are presented in the following sections.

### *Image capture*

To capture images of the printing nozzle a webcam was used (Logitech C920 HD Pro Webcam), using commands provided by OpenCV a frame or an instant image of the video capture is stored, that can be read later. By default, the size of the image is 640x480 but it can be modified if desired. The code lines used for the video capture were:

```
cap = cv2.VideoCapture(0)
// 0 is the assigned position of the camera.
ret, frame = cap.read()
```

### *Determining the color of interest*

The image captured has a BGR color space by default (RGB but in different order). In order to monitor the color of interest, the image is converted to the HSV color space, this way, the color information is separated from the brightness of the image. The conversion process is done by the OpenCV commands cvtColor and COLOR_BGR2HSV that transforms the input BGR image (RGB) to HSV as follows:

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Having the image in HSV allows to determine color content regardless of variations in illumination or saturation (up to some extent) and since the filaments to extrude are made of discrete basic colors, some ranges are established, making a more robust detection system. The chromatic circle can be seen in **Figure 2**, where the ranges are evident. To establish the HSV parameter ranges an online tool named "Colorizer.org" was used; this tool allows and easy conversion between space colors. HSV ranges were defined for 10 different colors and can be seen in Table 1.

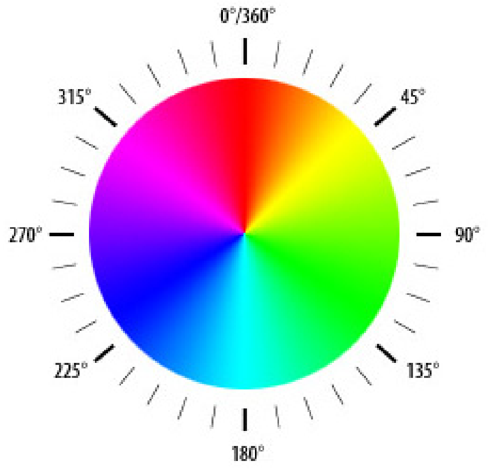


***Figure 2.*** *Chromatic circle with angles defined*

**Table 1. Ranges and limits of the HSV parameters for each filament color**

| Color | HSV/B (Hue, Saturation y Value/Brightness) | | | |
|---|---|---|---|---|
| | H (°) | H (OpenCV) | S (OpenCV) | V (OpenCV) |
| Red | 348 –359 & 0 – 4 | 174 – 179 & 0 – 2 | 120 – 255 & 160 – 255 | 125 – 255 |
| Orange | 20 – 40 | 10 – 20 | 160 – 255 | 125 – 255 |
| Yellow | 44 – 70 | 22 – 35 | 160 – 255 | 125 – 255 |
| Green | 90 – 140 | 45 – 70 | 150 – 255 | 80 – 255 |
| Cyan | 164–194 | 82 – 97 | 160 – 255 | 125 – 255 |
| Blue | 214–240 | 107 – 120 | 100 – 255 | 70 – 255 |
| Violet | 254–278 | 127 – 139 | 40 – 255 | 10 – 255 |
| Magenta | 284–330 | 142 – 165 | 40 – 255 | 50 – 255 |
| Black | 0 – 359 | 0 – 179 | 0 – 65 | 0 – 65 |
| White | 0 – 359 | 0 – 179 | 0 – 30 | 170 – 255 |

In Table 1, there are two columns for each Hue value, the reason is because the values in the chromatic circle (H °) goes from 0° to 360°, but OpenCV uses a range value of 0 to 180. Similarly, the saturation and brightness values are normally in a range of 0 to 100, but OpenCV uses a range of values from 0 to 255.

The color red is a special case that has two ranges because of its location in the chromatic circle, before 360° and after 0°; defining two ranges allows to fully detect the different variations of the color red.

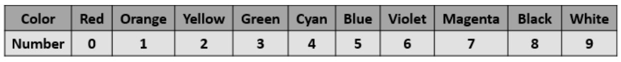
The ranges of each color are defined through a vector of the upper limit and other for the lower limit using the command "np.array()", where the first position corresponds to Hue, the second to saturation and the third to brightness. The code lines have the following form:

```
Color_LowerLimit = np.array([Hue_LowerLimit, Saturation_LowerLimit, Bright_LowerLimit])
Color_UpperLimit = np.array([Hue_UpperLimit, Saturation_UpperLimit, Bright_UpperLimit])
```

An example applied to one of the colors, in this case green:

```
Green_LowerLimit = np.array([45, 150, 80])
Green_UpperLimit = np.array([70, 255, 255])
```

The interface allows the user to know the different colors available for the filament detection and to choose the color that he wishes to detect. Each color is represented by one number (see **Figure 3**), therefore when the user types the number corresponding to the color he wishes to detect, the program will start automatically the detection and it will display a window showing the captured image and the binarized image in real time.



| Color | Red | Orange | Yellow | Green | Cyan | Blue | Violet | Magenta | Black | White |
|---|---|---|---|---|---|---|---|---|---|---|
| Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

***Figure 3.*** *Color options and its corresponding numbers.*

## Monitoring Material Flow

Once the color of the filament has been defined, the image is processed to extract only the material flow out of the nozzle, from the background and other elements that may appear. The process is based on performing a binarization of the image according to the color of interest on a region of interest.

### *Image binarization*

Image binarization is an image processing technique that reduces the information of a digital image to two Boolean values (True and false) or also represented as 0 (Black) and 255 (White). The binarization process consists in comparing each pixel of the image with a defined reference value or range. Therefore, the pixels that satisfy the reference value or range condition (Over/Below the reference value or In/Out of the reference range) will have a value of 255 (white pixels), while the pixels that don't satisfy the condition will have a value of 0 (black pixels).

The ranges defined before for each color are used as the condition to binarize the image. A black and white image is obtained as a result, where the white pixels correspond to the color of interest that is being detected and the black pixels are any other color that is not of interest.

The command that allows the binarization using as reference values the ranges of the chromatic circle is "cv2.inrange()". It has as input the HSV image that was converted before from RGB, and the upper and lower limits of the reference range.

```
BinarizedImage = cv2.inRange(ImageHSV, Color_LowerLimit, Color_UpperLimit)
```

An example applied to green filament can be seen below and its result in **Figure 4**.

```
mask = cv2.inRange(hsv, Green_LowerLimit, Green_UpperLimit)
```

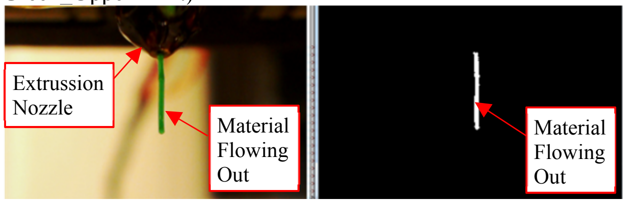


***Figure 4.*** *Example of binarization (Left: Original image, Right: Binarized image for green filament detection)*

### *Specify a region of interest*

To reduce the probability of external objects interfering with the data capture and causing an undesired alteration on the result of the image processing, a region of interest is defined, where the pixels of the image that are located out of the region are not taken into account. To define the region of interest in Python first a rectangle with the dimensions of the input image was made:

```
RectangleImage = np.zeros(frame.shape[:2], dtype = 'uint8')
```

And then the polygon that represent the region of interest was made using the next line code:

```
cv2.rectangle(RectangleImage, (x1,y1), (x2,y2), (255,255,255), -1)
```

The command "cv2.rectangle" allows to draw the function of a rectangle, it has as input parameters the first point of the rectangle vertex "(x1,y1)" and the second point that is the opposite vertex of the first point "(x2,y2)", the next parameter is the color of the rectangle in this case the colors was defined as white in the RGB space color "(255,255,255)" and finally the parameter that controls the thickness of the rectangle lines (if value is positive) or if is a fill rectangle (if the value is negative), in this case a white filled rectangle was defined so this parameter was assigned as a negative value "-1". This way the rectangle defined act as a mask to the image, blocking all the pixels on the frame except for the ones inside the rectangle. An example of defining the region of interest can be seen in **Figure 5**.

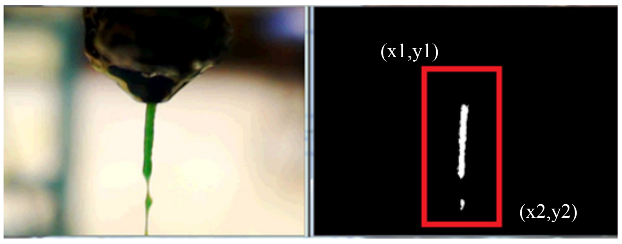


***Figure 5.*** *Rectangular region of interest*

### *Continuous detection*

The supervisory system relies on processing each one of the images or frames that the camera captures; therefore, a real time detection (or almost real time) is necessary for the program to work in a continuous way until the stop order is given.

An infinite "while" cycle was implemented, that means that the process of detection of filament color will be performed for each image captured by the webcam and in real time is analyzed. If the user wants to exit the infinite loop because he wants to change the color that will be detected or just to shut off the program, the next code line in Python will allow it:

```
if cv2.waitKey(1) & 0xFF == ord ('q'):
break
cv2.destroyAllWindows()
```

The command cv2.waitkey waits a time (in milliseconds) for a key to be pressed, in this case the letter "q" key. When the key is pressed the program exits the while infinite loop and destroys all the image windows that the program has displayed. The structure of the program has the following form:

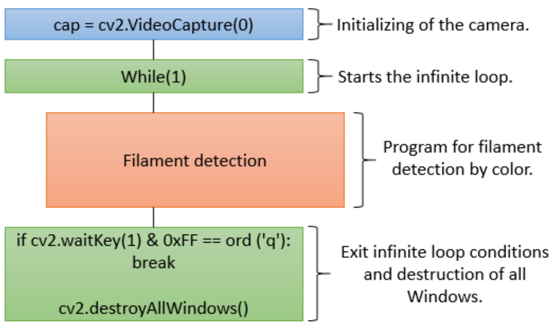

cap = cv2.VideoCapture(0)
Initializing of the camera.
While(1)
Starts the infinite loop.
Filament detection
Program for filament detection by color.
if cv2.waitKey(1) & 0xFF == ord ('q'): break
cv2.destroyAllWindows()
Exit infinite loop conditions and destruction of all Windows.

***Figure 6.*** *Program structure for continuous detection*

When the user wants to change the color of the filament to detect, he will press the key "q" and automatically the windows that displays the original and binarized image will be closed. Then the program will ask the user if he wishes to continue running the program and will wait for a "Y/N" answer. If the user answers "Yes" the program will show again the filament color options and will wait for the user to type the corresponding number for the desired color, otherwise the program ends.

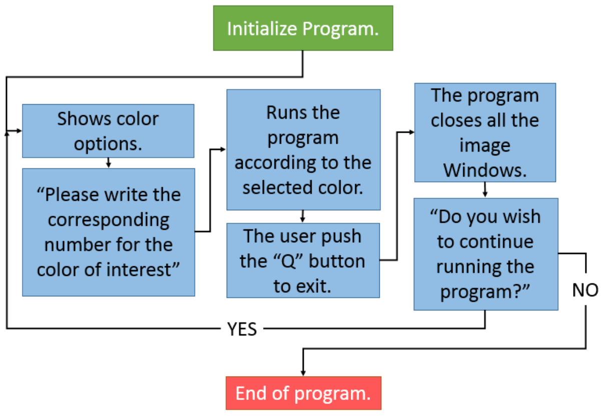

Initialize Program.
Shows color options.
"Please write the corresponding number for the color of interest"
Runs the program according to the selected color.
The user push the "Q" button to exit.
The program closes all the image Windows.
"Do you wish to continue running the program?"
NO
YES
End of program.

***Figure 7.*** *Program interface behavior with user.*

## Results

The first tests of the system were done with the extruder placed in a fixed position. Four different colors of filament were used: Red, Green, White, and Black, having successful detections for all four. Detection for the red and green filament was satisfactory; it can be observed clearly the discontinuities on the exit of material flow caused by the absence or tangled/blocked filament.

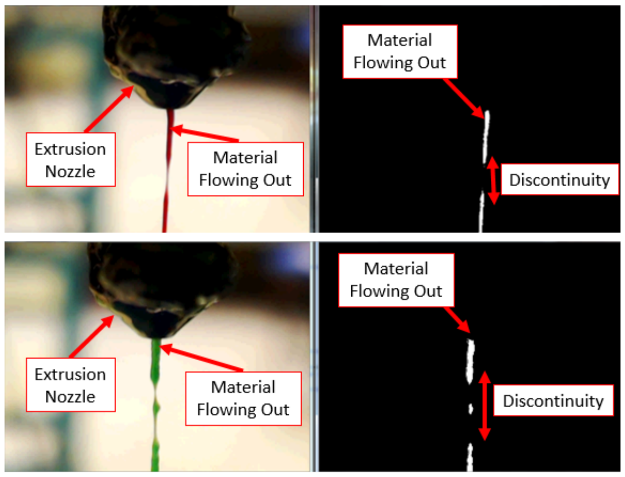


***Figure 8.*** *3D printing error detection (Top: Red filament, Bottom: Green filament)*

The white and black filament are special cases because, depending on the background, the filament can be more difficult to detect due to the lack of a specific range to look for in the chromatic circle, mimicking with other elements of the background. To solve this problem, a background with a color that would not interfere with any of the color options had to be placed. In the tests, a black background was used to detect the white filament, and a white background was used to detect the black filament. The detection of white and black color filament was satisfactory and the discontinuities can be observed, caused by the absence or tangled/blocked filament (see **Figure 9**).

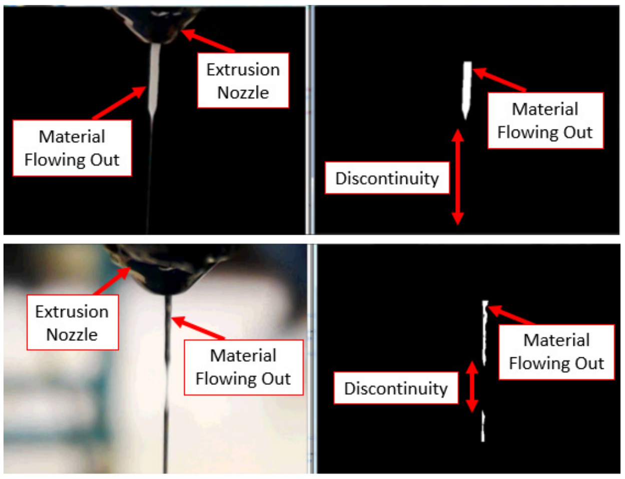


***Figure 9.*** *3D printing error detection (Top: White filament, Bottom: Black filament)*

Similar results have been achieved with all 10 of the color choices programed in the interface developed. Other errors were detected while allowing for material to flow with the extruder at a stationary position. For example, it was detected that sometimes the extruded material does not flow straight down but sticks to the outside of the extrusion nozzle, forming loops and blobs of material (see **Figure 10**). Although this is more notorious when the printing platform is far from the extrusion nozzle, it also happens during the printing process and it can cause problems in the deposition of filament at the intended location or for the following layers. As a consequence, the final printed object can be ruined or have surface defects that do not reproduce well the intended geometry.

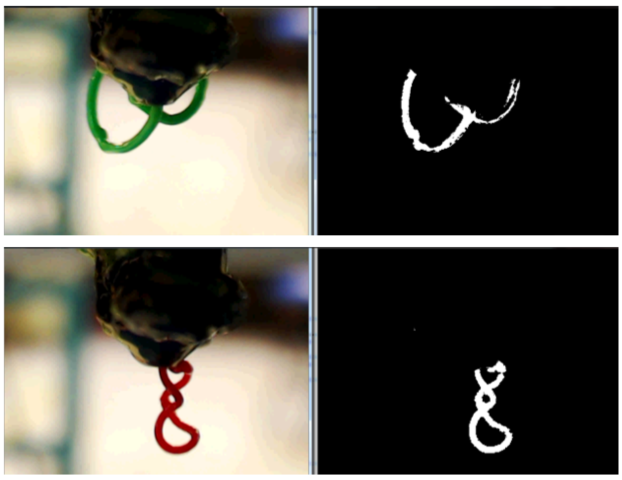


***Figure 10.*** *3D printing error caused by extruded material sticking out the extrusion nozzle.*

The supervisory system was tested during a 3D printing process of a callibration cube (dimensions of each side = 3cm). Two different 3D printing errors were detected: The first error was because the material did not adhere correctly to the surface of the building platform; therefore, an accumulation of material is generated outside de extrusion nozzle (see **Figure 11**) that can interrupt de deposition of following layers and ruin the final printed object, as it was noted for the previous tests. The second error was caused by the undesired flow of material. This excessive flow can cause a malformation of the layers that are being deposited, resulting in a defective print or the inaccurate reproduction of the intended geometry (see **Figure 12**).

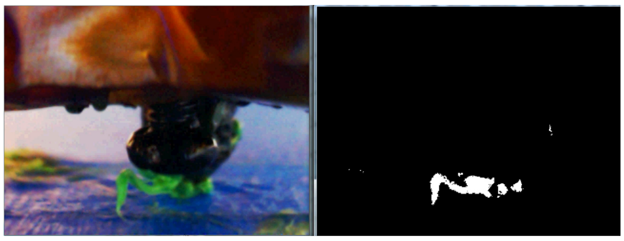


***Figure 11.*** *3D printing error caused by the material that didn't adhere correctly to the surface of the building platform.*

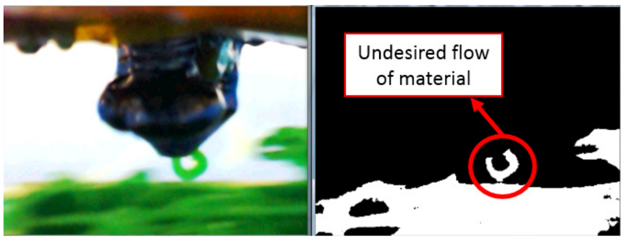


***Figure 12.*** *3D printing error caused by the excessive undesired flow of material.*

More complex tests are currently underway, particularly to achieve the detection without disrupting the printing process or the partially printed structure with the camera. Vibrations and other interference are also being considered in order have reliable detection of the material flow out of the nozzle.

## Conclusions

The work reported illustrates the first step towards closing the control loop by supervising the material output in the printing process. This is still a work in progress, however, preliminary results are promising, the detection has been successful for the 10 colors that were selected and that correspond to the more common filament colors used in FFF 3D printers. The system is able to detect the material output and discontinuities in the flow that can be correlated to the printing process to determine whether is intended or not.

One of the more challenging aspects to overcome is placing the camera to visualize well the material flow while printing without interfering with the printing process or the partially printed structure. The intent of the project has been to develop the supervising system with low cost components; therefore, other alternatives of low cost cameras are being evaluated with a small form factor or capturing the image indirectly.

Additionally, the detection process is being performed in an embedded system (Raspberry Pi) in real time, which shows that the image processing technique does not require excessive resources to achieve the desired results.

Once the detection of the discontinuities in the flow are detected, the printing process can be paused to manually fix the issues causing the disruption; however, pausing and resuming the printing process can occur only when an instruction (or a few instructions) of the G-code has been completed. This is not ideal since the interruption can happen in the middle of the trajectory of the instruction and even with the detection, pause and resume of the printing process, there would be a gap in the layer that may prevent an accurate reproduction of the intended geometry. A more robust control system would be needed that includes feeding back the position of each axis to determine the location of the printhead and the platform at each moment during the printing process.

## References

[1] P. J. Bartolo, Stereolithography - Materials, Processes and Applications: Springer, 2011.

[2] S. Kalpakjian and S. R. Schmid, *Manufactura, ingeniería y tecnologia*, 4 ed. Mexico: Pearson Education, 2002.

[3] I. Gibson, D. W. Rosen, and B. Stucker, *Additive Manufacturing Technologies - Rapid Prototyping to Direct Digital Manufacturing*. New York Springer, 2010.

[4] D. L. Bourell, M. C. Leu, and D. W. Rosen, "Roadmap for Additive Manufacturing Identifying the Future of Freeform Processing," ed. Austin, TX: The University of Texas at Austin, 2009.

[5] S. Blandon, J. C. Amaya, and A. J. Rojas, "Development of a 3D Printer and a Supervision System Towards the Improvement of Physical Properties and Surface Finish of the Printed Parts," presented at the 2nd Colombian Conference on Automatic Control, Manizales, Colombia, 2015.

[6] A. J. Rojas Arciniegas, "Towards the control of electrophotographic-based 3-dimensional printing: Image-based sensing and modeling of surface defects," Imaging Science Ph.D., Chester F. Carlson Center for Imaging Science, College of Science, Rochester Institute of Technology, Rochester, NY, 2013.

[7] A. J. Rojas Arciniegas, M. Esterman, and J. C. Cockburn, "Towards the control of the EP3D printed surface," *ASME Journal*

*of Manufacturing Science and Engineering,* vol. 137, pp. 021012-1 - 10, 2015.

## Author Biography

*Alvaro J. Rojas Arciniegas is an assistant professor at Universidad Autonoma de Occidente (UAO) in Cali, Colombia. He holds a PhD in Imaging Science from the Chester F. Carlson Center for Imaging Science of Rochester Institute of Technology (RIT), MS degrees in Industrial Engineering from RIT and in Systems and Entrepreneurial Engineering from University of Illinois at Urbana-Champaign, and a BS in Mechatronic Engineering from UAO. His research interests include 3D Printing, System Modeling, Product and Process Design Methodologies, Control, and Image Processing. He has combined his academic experience with industry work developing projects of technological improvement and innovation.*

*Juan C. Amaya Hurtado is a mechatronic engineering student and also a student member of the research group in technology for manufacture (GITEM) at Universidad Autonoma de Occidente. He is currently in the last semester of his program, working in the design of a supervision system for the extrusion of polymeric materials in open source 3D printers. His interests include 3D printing, Control, Image Processing, Design and product development methods.*